ソフトウェア信頼性評価のための確率微分方程式モデルと最適リリース問題への応用 (不確実性と意思決定数理の諸問題)

AUTHOR(S):

井上, 真二; 西垣, 陽夫; 山田, 茂; 木村, 光宏

CITATION:

井上, 真二 ...[et al]. ソフトウェア信頼性評価のための確率微分方程式モデルと最適リリース問題への応用 (不確実性と意思決定数理の諸問題). 数理解析研究所講究録 2004, 1373: 80-88

ISSUE DATE:

2004-05

URL:

http://hdl.handle.net/2433/25529

数理解析研究所講究録 1373 巻 2004 年 80-88

ソフトウェア信頼性評価のための確率微分方程式モデルと最適リリース問題への応用

鳥取大学大学院工学研究科　井上 真二 (Shinji Inoue)[†]
鳥取大学大学院工学研究科　西垣 陽夫 (Akio Nishigaki)[†]
鳥取大学　山田 茂　(Shigeru Yamada)[††]
法政大学　木村 光宏 (Mitsuhiro Kimura)[†††]

[†]Graduate School of Engineering, Tottori University
[††]Tottori University
[†††]Hosei University

1　はじめに

ソフトウェア信頼度成長モデル（software reliability growth model，以下 SRGM と略す）[3, 9, 10] は，現在，ソフトウェアシステムの定量的な信頼性評価手法の基盤技術の 1 つとして浸透している．特に，フォールト発見事象を離散状態空間上で取り扱う計数過程である非同次ポアソン過程（nonhomogeneous Poisson process，以下 NHPP と略す）に基づく SRGM は，比較的単純なモデル構造と適用性の高さから，多くの企業において実用にも供されている SRGM の 1 つであり，その適用事例 [2, 4] も数多く報告されている．

一方，近年では，ソフトウェアシステムの大規模化の傾向が強まる中，フォールト発見事象を連続状態空間上で取り扱う連続状態型 SRGM[8, 12] も提案されている．これらのモデルの多くは，対象とするソフトウェアシステムが大規模である場合，そのテスト工程において検出されるフォールト数も大きくなり，個々のデバッグ作業により修正・除去されるフォールト数の変化量は，テスト終了までに作り込まれた総フォールト数に比べて十分小さくなるという考え方から構築されたモデルである．連続状態型 SRGM に関して，これまでに，対数正規過程に基づいたものや [7, 8, 12]，ソフトウェアの逐次的品質評価への適用を念頭に入れたオルンシュタイン・ウーレンベック過程（Ornstein-Uhlenbeck process）に基づくもの [1] など，連続状態型 SRGM を構築するための枠組みが提案されている．しかしながら，今までの議論では，SRGM における状態空間の連続化手法だけに焦点が当てられ，ソフトウェアの信頼度成長過程を特徴付ける 1 個当りフォールト発見率に対して具体的な関数を与えた場合の詳細な議論は，ほとんどなされていない．したがって，これが実用に供されるようになるためには，それぞれのモデルに含まれる 1 個当りのフォールト発見率の時間的挙動を具体的に考慮した連続状態型 SRGM に関する議論が必要である．

本論文では，対数正規過程に基づく連続状態型 SRGM の構築の枠組みに基づいて，このモデルに含まれる 1 個当りのフォールト発見率の時間的挙動を表す関数を与えることにより，新しい SRGM を構築する．具体的には，1 個当りのフォールト発見率の時間的挙動を表す関数として，代表的な NHPP モデルである指数形 SRGM，遅延 S 字形 SRGM，および習熟 S 字形 SRGM の基本的仮定から導出される 1 個当りのフォールト発見率を適用する．これにより，連続状態型 SRGM としての，指数形，遅延 S 字形，および習熟 S 字形確率微分方程式モデルをそれぞれ構築する．また，本論文では，ソフトウェア開発管理面からの興味ある問題の 1 つとして，これらモデルを用いたソフトウェアの最適リリース（出荷）問題についても議論する．

2　連続状態型フトウェア信頼度成長モデル

本章では，対数正規過程に基づく連続状態型 SRGM[8, 12] の構築の枠組みについて簡単に議論すると共に，その解過程に含まれる 1 個当りのフォールト発見率の時間的挙動を表す関数を具体的に与えることにより，新しい SRGM を構築する．

まず，テスト時刻 t までに発見される総フォールト数を $N(t)$ とする．但し，$N(t)$ は，連続な実数値をとりながら変化する確率過程として考える．この $N(t)$ の時間的挙動は，以下のような微分方程式で記述されるものとする．

$$\frac{dN(t)}{dt} = b(t)\{a - N(t)\} \qquad (a > 0,\ b(t) > 0). \tag{1}$$

ここで，a はテスト開始前にソフトウェア内に潜在する総フォールト数，$b(t)$ はテスト時刻 t における 1 個当りのフォールト発見率を表す．

デバッグ作業は，作業者のフォールト発見能力や作業環境などの様々な要因に影響されながら複雑に進行していく．したがって，実際のテスト工程におけるフォールト検出過程は，確率的要因に大きく左右される．このことは，対象とするソフトウェアシステムが大規模になるにつれて顕著になると考えられる．そこで，テスト工程がもつ確率的要因を式 (1) に反映させるため，式 (1) に含まれる係数関数，つまり，テスト時刻 t における 1 個当りのフォールト発見率 $b(t)$ を時間的な不規則変動をもつように拡張する．すなわち，$\xi(t)$ を時間的に不規則に変動する雑音として，式 (1) を以下のような確率微分方程式 [6] に拡張する．

$$\frac{dN(t)}{dt} = \{b(t) + \xi(t)\}\{a - N(t)\}. \tag{2}$$

ここで，雑音 $\xi(t)$ の形としては様々なものが考えられるが，ここでは，解過程のマルコフ性を保証すること，および Itô の公式の適用を視野に入れ，以下のように表されるものとする．

$$\xi(t) = \sigma\gamma(t) \qquad (\sigma > 0). \tag{3}$$

ここで，σ は不規則変動の大きさを表す定数パラメータ，$\gamma(t)$ は標準化された白色雑音（Gauss 型白色雑音）である．式 (3) を式 (2) に代入することにより，

$$\frac{dN(t)}{dt} = \{b(t) + \sigma\gamma(t)\}\{a - N(t)\}, \tag{4}$$

を得る．式 (4) を Itô 型確率微分方程式 [6] に変換すると，

$$dN(t) = \{b(t) - \frac{1}{2}\sigma^2\}\{a - N(t)\}dt + \sigma\{a - N(t)\}dW(t), \tag{5}$$

のようになる．ここで，$W(t)$ は 1 次元ウィナー過程（one-dimensional Wiener process）を表し，形式的には，白色雑音 $\gamma(t)$ の時間積分に相当するものであり，以下のような性質をもつ．

(a) $\Pr[W(0) = 0] = 1$

(b) $\mathrm{E}[W(t)] = 0$

(c) $\mathrm{E}[W(t)W(t')] = \min[t, t']$

次に，解過程 $N(t)$ を導出する．解過程の導出方法としては，Itô の公式を用いる方法 [6]，Kolmogorov の前進方程式を用いる方法が挙げられるが，本論文で用いる確率微分方程式は線形であるため，前者を適用する．Itô の公式を用いることにより，式 (5) の解過程は，

$$N(t) = a\left[1 - \exp\left\{-\int_0^t b(s)ds - \sigma W(t)\right\}\right], \tag{6}$$

のように導出される．また，解過程 $N(t)$ の推移確率分布は，ウィナー過程 $W(t)$ がもつ性質である上記の (a)–(b) およびウィナー過程がガウス過程（Gaussian process）であることより，最終的に以下のように導出できる．

$$\Pr[N(t) \leq n | N(0) = 0] = \Phi\left(\frac{\log\frac{a}{a-n} - \int_0^t b(s)ds}{\sigma\sqrt{t}}\right). \tag{7}$$

ここで，$\Phi(\cdot)$ は，標準正規分布関数を表し，

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} \exp(-\frac{y^2}{2})dy, \tag{8}$$

と定義されるものである．

本論文では，式 (6) に含まれる 1 個当りのフォールト発見率 $b(t)$ に対して，具体的な関数を与えることにより，指数形，遅延 S 字形，および習熟 S 字形確率微分方程式モデルをそれぞれ構築する．

2.1 指数形確率微分方程式モデル

指数形確率微分方程式モデルは，従来の代表的なNHPPモデルである指数形SRGMの基本的仮定から，解過程である式(6)に含まれる1個当りのフォールト発見率を表す関数$b(t)$を次のように与えることにより導出される．

$$b(t) \equiv b_e(t) = b. \tag{9}$$

式(9)を式(6)に代入することにより，指数形確率微分方程式モデルの解過程$N_e(t)$は，

$$N(t) \equiv N_e(t) = a\left[1 - \exp\{-bt - \sigma W(t)\}\right], \tag{10}$$

と導出される．次に，指数形確率微分方程式モデルに対する解過程$N_e(t)$の推移確率分布を求める．これは，式(9)を式(7)に代入することにより，

$$\Pr[N_e(t) \leq n | N_e(0) = 0] = \Phi\left(\frac{\log\frac{a}{a-n} - bt}{\sigma\sqrt{t}}\right), \tag{11}$$

のように求められる．

2.2 遅延S字形確率微分方程式モデル

遅延S字形確率微分方程式モデルは，従来の代表的なNHPPモデルである遅延S字形SRGMの基本的仮定から，解過程である式(6)に含まれる1個当りのフォールト発見率を表す関数$b(t)$を次のように与えることにより導出される．

$$b(t) \equiv b_d(t) = \frac{b^2 t}{1 + bt}. \tag{12}$$

式(12)を式(6)に代入することにより，遅延S字形確率微分方程式モデルの解過程$N_d(t)$は，

$$N(t) \equiv N_d(t) = a\left[1 - (1 + bt)\exp\{-bt - \sigma W(t)\}\right], \tag{13}$$

と導出される．また，遅延S字形確率微分方程式モデルに対する解過程$N_d(t)$の推移確率分布は，指数形確率微分方程式モデルの場合と同様にして，次のように求められる．

$$\Pr[N_d(t) \leq n | N_d(0) = 0] = \Phi\left(\frac{\log\frac{a}{a-n} - bt + \log(1 + bt)}{\sigma\sqrt{t}}\right). \tag{14}$$

2.3 習熟S字形確率微分方程式モデル

習熟S字形確率微分方程式モデルは，従来の代表的なNHPPモデルである習熟S字形SRGMの基本的仮定から，解過程である式(6)に含まれる1個当りのフォールト発見率を表す関数$b(t)$を次のように与えることにより導出される．

$$b(t) \equiv b_i(t) = \frac{b}{1 + c \cdot \exp(-bt)}. \tag{15}$$

式(15)を式(6)に代入することにより，習熟S字形確率微分方程式モデルの解過程$N_i(t)$は，

$$N(t) \equiv N_i(t) = a\left[1 - \frac{1 + c}{1 + c \cdot \exp(-bt)}\exp\{-bt - \sigma W(t)\}\right], \tag{16}$$

と書き換えられる．また，習熟S字形確率微分方程式モデルに対する解過程$N_i(t)$の推移確率分布は，次のように求められる．

$$\Pr[N_i(t) \leq n | N_i(0) = 0] = \Phi\left(\frac{\log\frac{a}{a-n} - bt - \log\frac{1 + c \cdot \exp(-bt)}{1 + c}}{\sigma\sqrt{t}}\right). \tag{17}$$

3 ソフトウェア信頼性評価尺度

本章では，第2章において提案した3つのタイプの連続状態SRGMに基づいて，定量的なソフトウェア信頼性評価に有用なソフトウェア信頼性評価尺度を導出する．但し，ウィナー過程 $W(t)$ の確率密度関数 $f(W(t))$ は，ウィナー過程 $W(t)$ の性質である前述の (a)-(b) と $W(t)$ がガウス過程であることから，

$$f(W(t)) = \frac{1}{\sqrt{2\pi t}} \exp\left\{-\frac{W(t)^2}{2t}\right\}, \tag{18}$$

のように与えられる．

3.1 累積発見フォールト数の期待値

ここでは，第2章で導出した3つの確率微分方程式モデルに関して，累積発見フォールト数の期待値，すなわち，テスト時刻 t までに発見された総フォールト数の期待値をそれぞれ求める．まず，指数形確率微分方程式モデルを用いた場合，テスト時刻 t までに発見される総期待フォールト数は，

$$\begin{aligned} \mathrm{E}[N_e(t)] &= \mathrm{E}[a\{1-\exp(-bt-\sigma W(t))\}] \\ &= a\left\{1-\exp\left(-bt+\frac{\sigma^2}{2}t\right)\right\}. \end{aligned} \tag{19}$$

同様にして，遅延S字形および習熟S字形確率微分方程式モデルに関しては，それぞれ，

$$\mathrm{E}[N_d(t)] = a\left\{1-(1+bt)\exp\left(-bt+\frac{\sigma^2}{2}t\right)\right\}, \tag{20}$$

$$\mathrm{E}[N_i(t)] = a\left\{1-\frac{1+c}{1+c\cdot\exp(-bt)}\exp\left(-bt+\frac{\sigma^2}{2}t\right)\right\}, \tag{21}$$

と導出される．

3.2 発見フォールト数の分散

指数形，遅延S字形，および習熟S字形確率微分方程式モデルに関して，テスト時刻 t までに発見される総フォールト数の分散は，それぞれ，

$$\begin{aligned} \mathrm{Var}[N_e(t)] &= \mathrm{E}[(N_e(t)-\mathrm{E}[N_e(t)])^2] \\ &= a^2\exp\{-2bt+\sigma^2 t\}(\exp\{\sigma^2 t\}-1), \end{aligned} \tag{22}$$

$$\mathrm{Var}[N_d(t)] = a^2(1+bt)^2\exp\{-2bt+\sigma^2 t\}(\exp\{\sigma^2 t\}-1), \tag{23}$$

$$\mathrm{Var}[N_i(t)] = a^2\left\{\frac{1+c}{1+c\exp\{-bt\}}\right\}^2\exp\{-2bt+\sigma^2 t\}(\exp\{\sigma^2 t\}-1), \tag{24}$$

と求められる．

3.3 瞬間MTBF

平均故障発生時間間隔（mean time between software failures，以下MTBFと略す）の代替的評価尺度の1つである瞬間MTBF（instantaneous MTBF）について議論する．瞬間MTBFは，簡単化のために，

$$MTBF_I(t) = \frac{dt}{\mathrm{E}[dN(t)]}, \tag{25}$$

のように近似的に計算することにする．

式 (25) を用いて，指数形，遅延S字形，および習熟S字形確率微分方程式モデルの瞬間MTBFは，そ

れぞれ，次のように求められる．

$$MTBF_I^e(t) = \frac{1}{a(b-\frac{1}{2}\sigma^2)\exp\{-(b-\frac{\sigma^2}{2})t\}}, \tag{26}$$

$$MTBF_I^d(t) = \frac{1}{a(1+bt)(\frac{b^2t}{1+bt}-\frac{1}{2}\sigma^2)\exp\{-(b-\frac{\sigma^2}{2})t\}}, \tag{27}$$

$$MTBF_I^i(t) = \frac{1}{\frac{a(1+c)}{1+c\cdot\exp(-bt)}(\frac{b}{1+c\cdot\exp(-bt)}-\frac{1}{2}\sigma^2)\exp\{-(b-\frac{\sigma^2}{2})t\}}. \tag{28}$$

但し，式(26)–式(28)の瞬間MTBFを求める際，ウィナー過程$W(t)$がもつ独立増分性および$\mathrm{E}[dW(t)]=0$に注意する必要がある．

3.4 累積MTBF

累積MTBF（cumulative MTBF）は，瞬間MTBFと同様，MTBFの代替的評価尺度の1つであり，実用に供されている信頼性評価尺度の1つでもある．本論文では，簡単化のために，累積MTBFを

$$MTBF_C(t) = \frac{t}{\mathrm{E}[N(t)]}, \tag{29}$$

として，近似的に計算する．これにより，3つのタイプの確率微分方程式モデルの累積MTBFは，それぞれ，

$$MTBF_C^e(t) = \frac{t}{a\left\{1-\exp\left(-bt+\frac{\sigma^2}{2}t\right)\right\}}, \tag{30}$$

$$MTBF_C^d(t) = \frac{t}{a\left\{1-(1+bt)\exp\left(-bt+\frac{\sigma^2}{2}t\right)\right\}}, \tag{31}$$

$$MTBF_C^i(t) = \frac{t}{a\left\{1-\frac{1+c}{1+c\cdot\exp(-bt)}\exp\left(-bt+\frac{\sigma^2}{2}t\right)\right\}}, \tag{32}$$

のように求められる．

4 未知パラメータの推定法

第2章で議論した指数形と遅延S字形確率微分方程式モデルの推移確率分布に含まれているパラメータa, b, およびσ，また，習熟S字形確率微分方程式モデルの推移確率分布に含まれているパラメータa, b, c, およびσは，一般には既知ではないため，実測データなどの利用可能なデータを使ってそれらの値を推定しなければならない．本論文では，未知パラメータを推定する方法として，最尤法(method of maximum-likelihood)を用いる．

テスト工程における観測データは，一般に$(t_j, n_j)(j=1,2,\cdots,K)$という形で与えられているものとする．ここで，$n_j$は，テスト時刻$t_j$までに発見された総フォールト数である．まず，確率過程$N(t)$のK次の同時確率分布を

$$P(t_1,n_1;t_2,n_2;\cdots;t_K,n_K) = \Pr[N(t_1)\le n_1, N(t_2)\le n_2,\cdots N(t_K)\le n_K | N(0)=0], \tag{33}$$

とし，その同時確率密度を

$$p(t_1,n_1;t_2,n_2;\cdots;t_K,n_K) = \frac{\partial^K P(t_1,n_1;t_2,n_2;\cdots;t_K,n_K)}{\partial n_1 \partial n_2 \cdots \partial n_K}, \tag{34}$$

とする．ここで，$N(t)$は連続値を取るのため，観測されたデータ(t_j, n_j)に対し，尤度関数を

$$l = p(t_1,n_1;t_2,n_2;\cdots;t_K,n_K), \tag{35}$$

と表す．さらに，

$$L = \log\, l, \tag{36}$$

により対数尤度関数 L を導出する．これより，例えば，習熟 S 字形確率微分方程式モデルにおける未知パラメータ a, b, c, および σ は，

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial b} = \frac{\partial L}{\partial c} = \frac{\partial L}{\partial \sigma} = 0, \tag{37}$$

の解として数値的に得ることができる．

5 ソフトウェアの最適リリース問題

テスト工程において，本論文で導出された確率微分方程式モデルを用いた定量的な信頼性評価を行った後に，ソフトウェア開発管理面からの興味ある問題の 1 つとして，ソフトウェアの最適リリース（出荷）問題 (optimal software release problem) [11] が挙げられる．このソフトウェアの最適リリース問題とは，テスト工程に要するテストコストと，テスト工程において修正・除去できなかったフォールトに起因するリリース後（運用段階）における保守コストとのトレードオフの関係を考慮し，所定の評価基準を設定することにより，ソフトウェアをテスト工程から運用段階へ移行させるのに最適なテスト時刻を求める問題である．

NHPP モデルに基づく従来のソフトウェアの最適リリース時刻の推定には，点推定的手法が多く用いられていたが，本論文では，第 2 章で提案した 3 つの確率微分方程式モデルがもつ性質を考慮して，区間推定的手法によるソフトウェアの最適リリース時刻の推定を試みる．この推定手法を適用することにより，所定の信頼区間の下での最適リリース時刻および総ソフトウェアコストの範囲を推定することができ，ソフトウェア開発管理者は，納期およびソフトウェアコストの見積りに関して，従来手法よりも柔軟かつ定量的に対応できることが期待される．ところで，本論文では，評価基準として，総ソフトウェアコストのみを考慮した最も単純な場合について議論する．

5.1 総ソフトウェアコストの定式化とその分布関数の導出

ソフトウェアの最適リリース問題を議論するにあたり，テスト工程および運用段階において費やされる総ソフトウェアコストの定式化が必要となる．そこで，以下のようなコストパラメータを定義する．

c_1 = 単位時間当りのテストコスト（$c_1 > 0$）．
c_2 = テスト工程において発見されるフォールト 1 個当りの修正コスト（$c_2 > 0$）．
c_3 = 運用段階において発見されるフォールト 1 個当りの保守コスト（$c_3 > 0$, $c_3 > c_2$）．

これにより，総ソフトウェアコストは，以下のように定式化できる．

$$Cost(N(t), t) = c_1 t + c_2 N(t) + c_3\{a - N(t)\}. \tag{38}$$

ここで，$N(t)$ は確率変数であるため，式 (38) の $Cost(N(t), t)$ も確率変数であることに注意する．まず，式 (11) を用いて，指数形確率微分方程式モデルに対するソフトウェアコストの分布関数 $Cost(N_e(t), t)$ を考える．まず，式 (38) より，

$$N_e(t) = \frac{c_1 t + ac_3 - Cost(N_e(t), t)}{c_3 - c_2}, \tag{39}$$

となる．式 (11) に式 (39) を代入して $Cost(N_e(t), t)$ の分布関数を求める．ここで，

$$C = -n(c_3 - c_2) + (c_1 t + ac_3), \tag{40}$$

とおくと，式 (11) における Φ の（）内の分子は，

$$\log\left(\frac{a}{a - \frac{-C + c_1 t + ac_3}{c_3 - c_2}}\right) - bt = \log\left[\frac{a(c_3 - c_2)}{C - (c_1 t + ac_2)}\right] - bt,$$

と変形される．したがって，$Cost(N_e(t), t)$ の分布関数は，

$$\Pr[Cost(N_e(t), t) \leq C] = 1 - \Phi\left(\frac{\log \frac{a(c_3 - c_2)}{C - (c_1 t + ac_2)} - bt}{\sigma\sqrt{t}}\right), \tag{41}$$

と導出される．但し，C には制約があり，log の中の分母が正になっている場合についてのみ分布が意味をもつことに注意しなければならない．ところで，総期待ソフトウェアコスト $\mathrm{E}[Cost(N_e(t),t)]$ は，

$$\begin{aligned}\mathrm{E}[Cost(N_e(t),t)] &= \mathrm{E}[c_1 t + c_2 N_e(t) + c_3\{a - N_e(t)\}] \\ &= c_1 t - (c_3 - c_2)a\left\{1 - \exp\left(-bt + \frac{\sigma^2}{2}t\right)\right\} + ac_3, \end{aligned} \tag{42}$$

と求められる．同様にして，遅延 S 字形および習熟 S 字形確率微分方程式モデルを用いた場合の総ソフトウェアコストの分布関数は，それぞれ，

$$\Pr[Cost(N_d(t),t) \le C] = 1 - \Phi\left(\frac{\log\frac{a(c_3-c_2)}{C-(c_1 t+ac_2)} - bt + \log(1+bt)}{\sigma\sqrt{t}}\right), \tag{43}$$

$$\Pr[Cost(N_i(t),t) \le C] = 1 - \Phi\left(\frac{\log\frac{a(c_3-c_2)}{C-(c_1 t+ac_2)} - bt - \log\frac{1+c*\exp(-bt)}{1+c}}{\sigma\sqrt{t}}\right), \tag{44}$$

と求められる．さらに，総期待ソフトウェアコストも，それぞれ，

$$\mathrm{E}[Cost(N_d(t),t)] = c_1 t - (c_3 - c_2)a\left\{1 - (1+bt)\exp\left(-bt + \frac{\sigma^2}{2}t\right)\right\} + ac_3, \tag{45}$$

$$\mathrm{E}[Cost(N_i(t),t)] = c_1 t - (c_3 - c_2)a\left\{1 - \frac{1+c}{1+c\exp(-bt)}\exp\left(-bt + \frac{\sigma^2}{2}t\right)\right\} + ac_3, \tag{46}$$

と求められる．

5.2 総ソフトウェアコストの信頼区間と最適リリース時刻の導出

本論文では，総ソフトウェアコストを表す関数を確率変数として扱っているため，総ソフトウェアコストの分布関数の値が $(1-0.01\alpha)/2$ となる総ソフトウェアコストと，$(1+0.01\alpha)/2$ となる総ソフトウェアコストを求め，ソフトウェアコストの α% 区間を求める．この区間を，ソフトウェアコストの α% 信頼区間ということにする．このとき，ソフトウェアコストの α% 信頼区間の上下限は，それぞれ，

$$\mathrm{UCL} = \mathrm{E}[Cost(N(t),t)] + \beta_1(t)\sqrt{\mathrm{Var}[Cost(N(t),t)]}, \tag{47}$$

$$\mathrm{LCL} = \mathrm{E}[Cost(N(t),t)] - \beta_2(t)\sqrt{\mathrm{Var}[Cost(N(t),t)]}, \tag{48}$$

と与えられる．ここで，UCL および LCL は，それぞれ，ソフトウェアコストの α% 信頼区間の上限および下限を示す．但し，$\beta_1(t)$ および $\beta_2(t)$ は，それぞれ，

$$\beta_1(t) = \frac{\exp(\frac{\sigma^2 t}{2}) - \exp(-\sigma\sqrt{t}\Phi^{-1}(\frac{1-0.01\alpha}{2}))}{\sqrt{\exp(\sigma^2 t)(\exp(\sigma^2 t)-1)}}, \tag{49}$$

$$\beta_2(t) = \frac{-\exp(\frac{\sigma^2 t}{2}) + \exp(-\sigma\sqrt{t}\Phi^{-1}(\frac{1+0.01\alpha}{2}))}{\sqrt{\exp(\sigma^2 t)(\exp(\sigma^2 t)-1)}}, \tag{50}$$

と表される．ここで，$\Phi^{-1}(\cdot)$ は，標準正規分布関数の逆関数である．

ところで，最適リリース時刻を T^* とすると，T^* は総期待ソフトウェアコストを最小にする時刻 $t = T^*$ を求めることで得られる．さらに，本論文では α% 信頼区間の上限 UCL と下限 LCL を，それぞれ最小にする時刻 $t = T_U^*$ および $t = T_L^*$ を求めることで，α% 信頼区間における最適リリース時刻の範囲を求めることができる．

6 実測データを用いた数値例

本章では，実際のテスト工程において観測されたフォールト発見数データを用いて，本論文で議論した指数形，遅延 S 字形，および習熟 S 字形確率微分方程式モデル，さらに，第 5 章において議論したソフトウェアの最適リリース問題に対する数値例を示す．本論文において使用した実測データは，テスト時間の測定単位が週であり，$(t_j, n_j)(j = 1, 2, \cdots, 19)$ からなる 19 組のデータ [5] である．

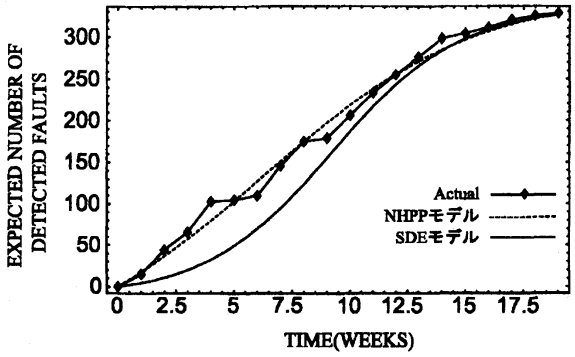

図 1： 推定された習熟 S 字形確率微分方程式モデルの累積発見フォールトの期待値，$\widehat{\mathrm{E}}[N_i(t)]$.

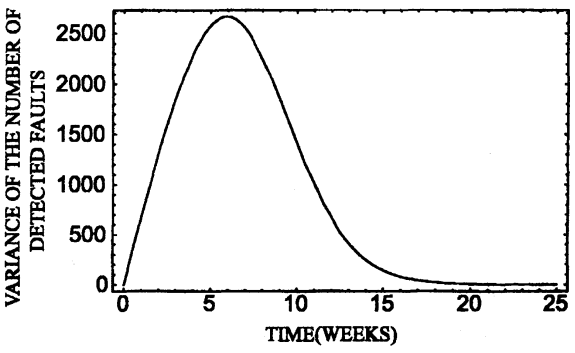

図 2： 推定された習熟 S 字形確率微分方程式モデルの累積発見フォールトの分散，$\widehat{\mathrm{Var}}[N_i(t)]$.

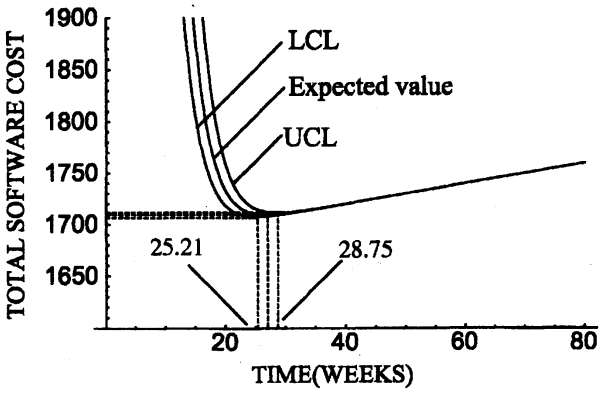

図 3： 習熟 S 字形確率微分方程式モデルによる総ソフトウェアコストの 90% 信頼区間.

6.1 ソフトウェア信頼性評価尺度に対する数値例

第 4 章で議論したパラメータ推定手法に基づいて，次のような最尤推定値 $\widehat{a}$, $\widehat{b}$, $\widehat{c}$, および $\widehat{\sigma}$ をそれぞれ以下のように得ることができた.

- 指数形確率微分方程式モデル：$\widehat{a} = 390.305$, $\widehat{b} = 0.0996$, $\widehat{\sigma} = 0.0561$
- 遅延 S 字形確率微分方程式モデル：$\widehat{a} = 349.449$, $\widehat{b} = 0.2370$, $\widehat{\sigma} = 0.07260$
- 習熟 S 字形確率微分方程式モデル：$\widehat{a} = 335.927$, $\widehat{b} = 0.360$, $\widehat{c} = 25.867$, $\widehat{\sigma} = 0.0784$

本論文では，習熟 S 字形確率微分方程式モデルについて数値例を示す．まず，習熟 S 字形確率微分方程式モデルに関して，推定された累積発見フォールト数の期待値 $\widehat{\mathrm{E}}[N_i(t)]$ および分散 $\widehat{\mathrm{Var}}[N_i(t)]$ を，それぞれ図 1 および図 2 に示す．図 1 より，任意のテスト時刻 t までに発見される総期待フォールト数の推定値は，実測データが示す成長曲線をうまく再現していることがわかる．一方，図 2 より，その分散は，テスト開始後しばらく経過した時点で最大値をとり，その後速やかに減少していくことがわかる．

6.2 ソフトウェアの最適リリース問題に対する数値例

次に，第 5 章において議論したソフトウェアの最適リリース問題に対して，その数値例を与える．ここでは，一例として，コストパラメータを

$$c_1 = 1,\ c_2 = 5,\ c_3 = 10,$$

のように設定した場合のソフトウェアの最適リリース時刻を求めることにする．ここでも，習熟 S 字形確率微分方程式モデルに焦点を当て，数値例を示していく．但し，習熟 S 字形確率微分方程式モデルのパラ

メータ推定値は，**6.1** で示したものを利用する．図 3 に，総ソフトウェアコストの 90% 信頼区間の変化を示す．図 3 より，$T_U^* = 28.75$(日) および $T_L^* = 25.21$(日) と推定された．

7 おわりに

本論文では，対数正規過程に基づく連続状態型 SRGM の構築の枠組みに基づいて，1 個当りのフォールト発見率の時間的挙動を表す関数を具体的に示すことにより，3 つのタイプの確率微分方程式モデルを構築した．さらに，ソフトウェア開発管理面からの興味ある問題の 1 つとして，これらモデルを用いたソフトウェアの最適リリース（出荷）問題についても議論した．また，ソフトウェアの最適リリース時刻を推定する際には，提案した 3 つの確率微分方程式モデルがもつ性質を考慮して，区間推定手法によるコスト評価基準によるソフトウェアの最適リリース時刻の推定を試みた．これにより，所定の信頼区間の下での最適リリース時刻および総ソフトウェアコストの範囲を推定することができ，ソフトウェア開発管理者は，納期およびソフトウェアコストの見積りに関して，従来手法よりも柔軟かつ定量的に対応できることが期待される．

参考文献

[1] 安藤 隆夫，土肥 正，岡村 寛之，“連続状態ソフトウェア信頼性モデル，” 2003 年日本オペレーションズ・リサーチ学会春季研究発表会アブストラクト集, pp. 34–35, 2003.

[2] K. Konoun, M. Kaaniche, and J.C. Laprie, “Qualitative and quantitative reliability assessment,” *IEEE Software Magazine*, Vol. 14, No. 2, pp. 77–87, 1997.

[3] J.D. Musa, D. Iannio, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[4] T. Nara, M. Nakata, and A. Ooishi, “Software reliability growth analysis —Application of NHPP models and its evaluation—,” *Proceedings of The 1996 High-Assurance Systems Engineering Workshop* (HASE'96), pp. 222–227, 1996.

[5] M. Ohba, “Software reliability analysis models,” *IBM J. Research and Development*, Vol. R-34, pp. 422–424, 1985.

[6] B. Øksendal, *Stochastic Differential Equations : An Introduction with Applications*, Springer-Verlag, Berlin, 1985.

[7] 田村 慶信，木村 光宏，山田 茂，“分散開発環境に対するソフトウェア信頼度成長モデル：確率微分方程式アプローチとその推定，” 日本応用数理学会論文誌, Vol. 11, No. 3, pp. 121–132, 2001.

[8] 田中 泰明，山田 茂，川上 真一，尾崎 俊治，“ソフトウェア信頼度成長モデルにおけるエラー数の状態空間の連続化に関する考察 — 線形確率微分方程式の適用 —，” 電子情報通信学会論文誌, Vol. J74-A, No. 7, pp. 1059–1066, 1991.

[9] 山田 茂，ソフトウェア信頼性モデル —基礎と応用—，日科技連出版社，東京，1994.

[10] S. Yamada, “Software reliability models,” in *Stochastic Models in Reliability and Maintenance* (S. Osaki, Ed.), Springer-Verlag, Berlin, pp. 253–280. 2002.

[11] S. Yamada and S. Osaki, “Cost-reliability optimal release policies for software systems,” *IEEE Trans. Reliability*, Vol. R-34, pp. 422–424, 1985.

[12] S. Yamada, M. Kimura, H. Tanaka, and S. Osaki, “Software reliability measurement and assessment with stochastic differential equations,” *IEICE Trans. Fundamentrals of Electronics, Communications and Computer Sciences*, Vol. E77-A, No. 1, pp. 109–116, 1994.